

# ARGUI 1

Aux. Jorge Gutierrez 900.g1 / 5Gqj3F

gojorge1@gmail.com

Lab Sabado 11-1

[ARGUI 1] DATOS

## Contenido:

### UNIDAD N°1: ARQUITECTURA CPU'S INTEL

- |         |   |
|---------|---|
| CAP.    | * ARQUITECTURA CPU BÁSICO (8 BITS) CISC / VON NEUMANN |
| Lb/Txt  | * ARQUITECTURA CPU INTEL 286                          |
| 1,2,3   | * ORGANIZACION DE REGISTROS DE CPU'S INTEL X86+P.I    |
| 1er     | * ADMINISTRACION DE MEMORIA POR SEGMENTACION          |
| Parcial | * MODOS DE DIRECCIONAMIENTO EN CPU'S INTEL            |

### UNIDAD N°2 : PROGRAMACION A BAJO NIVEL CPU'S INTEL

#### PLATAFORMA 8/16 BITS

- |        |   |  |
|--------|---|--|
| CAP    | * SET INSTRUCCIONES MOVIMIENTO DE INFORMACION |  |
| Lb/Txt | II  | II LÓGICAS                               |
| 4,5,6  | # II  | II ARITMÉTICAS                           |
|        | * II  | II SALTOS CONDICIONADOS/NO CONDICIONADOS |
|        | * II  | II DESPLAZAMIENTO Y ROTACIÓN             |
|        | * RETARDOS POR SOFTWARE                       |  |
|        | * SALIDA / ENTRADA A PUERTOS                  |  |
|        | * EJERCICIOS DE PROGRAMACIÓN                  |  |

S A O

### UNIDAD N°.3 : TÉCNICAS DE ATENCIÓN A PUERTOS

- \* POLLING
  - \* INTERRUPCIONES
  - \* TRASMISIÓN SERIE
  - \* EJERCICIOS DE PROGRAMACIÓN POR INTERRUPCIÓN EN SOFTWARE
- (AP. LIBTXT) 7, 11, 12  
3er Parcial

### UNIDAD N°.4 : MODO PROTEGIDO / BUSES / CISC VERSUS RISC

- \* ADMINISTRACIÓN DE MEMORIA EN MODO PROTEGIDO
  - \* BUSES
  - \* CISC VERSUS RISC.
- Cap lib TXT  
18 fotocopias anexas  
FINAL

### Bibliografía

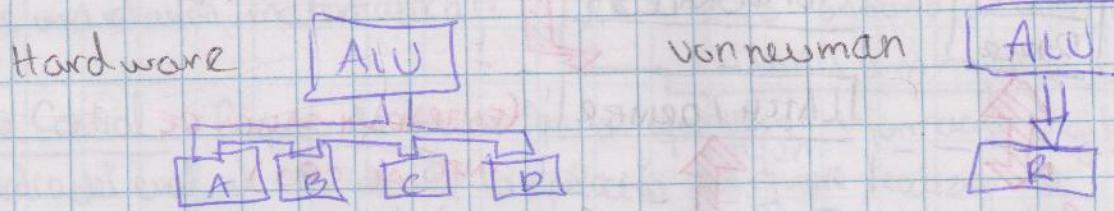
- Libro de Texto :
- \* Micro - procesadores INTEL
  - \* Barry B. Grey
  - \* PRENTICE - HALL
  - \* Esta 7ma y 8va Edición

30 pts de lab

$$14 \text{ c/parcial} * 3 = 42$$

3 cortos y Tareas

- \* La característica básica de los CPU's es que todos tienen un ALU.
- \* ALU (corazón del CPU)
- \* VON NEUMAN : Características
  - // No tienen capacidad para manejar un área para comando y una para dato separada
  - // Al desarrollar una operación en el ALU se depositaba en un solo registro (Registro Acumulador) el resultado.
  - // Los cpu's von neuman tienen una disposición NO óptima de buses internas, no tienen la capacidad de realizar eventos paralelo en tiempo real
- \* Con arquitectura hardware : Contrario del von neuman



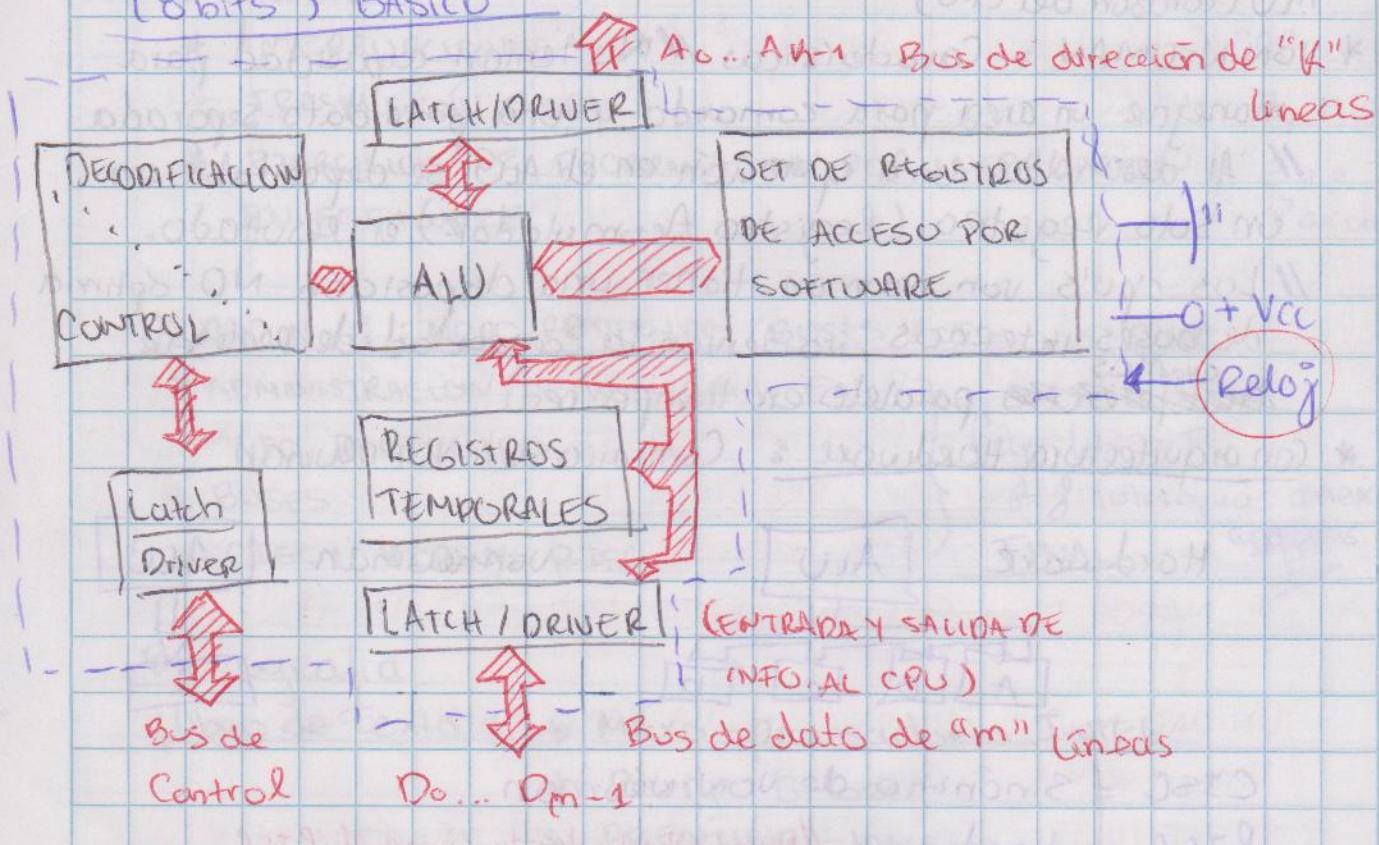
CISC = Sinónimo de VON Neuman

RISC = Hardware //No se cuenta tanto como el CISC

Línea | Arquitectura

- \* CISC - Poseen un software a bajo nivel con el objetivo de hacer un salto semántico corto.
- // Posee poca cantidad de registros internos
- / Calidad de buses no es buena

## VISTA A BLOQUES DE LA ARQUITECTURA DE UN CPU CISC/V.N (8 bits) BÁSICO



Se dice que el CPU es 100% secuencial aunque el ALU sea 100% combinacional.

### UNIDAD DE CODIFICACIÓN:

interpretar los códigos de operación que representen una orden o comando que se va a ejecutar.

**LATCH / DRIVER :** Memoriza info q se escribe en ella (latch).  
Drena corriente eléctrica afuera y adentro, maneja valores eléctricos de amperaje para que no doblen lo de adentro (Driver).

Buses: (Bus de Datos) Conjunto de líneas físicas que llevan información digital que fue o será computada por el CPU. Proporciona una calificación de índole absoluta. Del tamaño de los buses de datos (ancho) depende la calidad del CPU. Entre más ancho es mejor el CPU, porque puede aceptar cadenas binarias más anchas.

Bus de Dirección: Conjunto de líneas físicas que llevan información digital por medio de la cual el CPU identifica a una celda de memoria específica o puerto específico.

Existe una relación matemática:  $2^k = \# \text{ / } \# \text{ líneas}$  cantidad de memoria máxima a manejar en las celdas

Bus de Control: Conjunto de líneas físicas que llevan información digital por medio del cual el CPU decide la operación que quiere realizar la sincroniza y la temporiza.

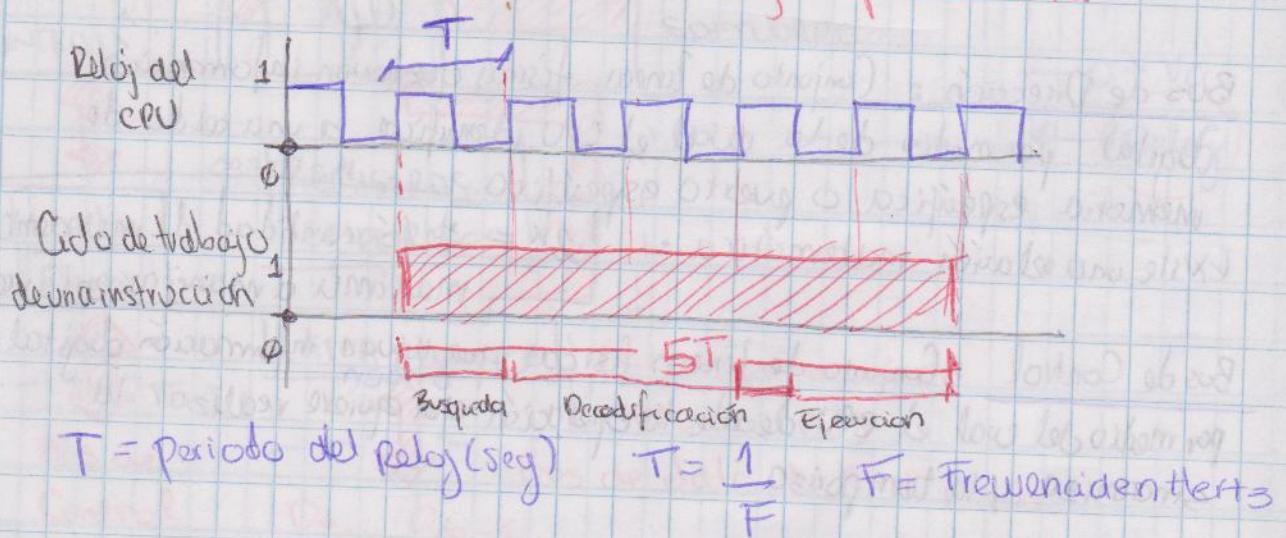
I/O REQUEST (Puertos)  $\times \rightarrow$  Solo se puede activar una  
MEM REQUEST (Memoria)  $\rightarrow \times$  a la vez

~~I/O REQUEST~~  $\leftarrow$  RO  $\rightarrow$  escribir en puerto ✓ Definir  
~~MEM REQUEST~~  $\leftarrow$  RO WR ✓ Sincronizar  
WR ✓ Temporizar

## Investigar (overclocking)

### CICLO DE TRABAJO ODE INSTRUCCIÓN DE UN CPU

Es el lapso de tiempo que le toma al CPU realizar una orden terminalmente de inicio a fin. Período o ciclo de reloj es la unidad de tiempo internacional. Es el tiempo que tarda el reloj en cambiar de un 0 a 1. Frecuencia nominal del reloj es el tiempo en el que el CPU trabaja optimamente.



#### \* SUB-CICLOS :

\* BUSQUEDA

\* DECODIFICACION

\* EJECUCION

\* Busquedas: Período o lapso de tiempo en el cual el CPU va a memoria a buscar el código de operación de la instrucción que debe desarrollar, lo lee, lo toma y se lo entrega a la unidad de decodificación.

\* Decodificación: Período o lapso de tiempo en el cual el CPU interpreta el código de operación en función de lo que debe desarrollar.

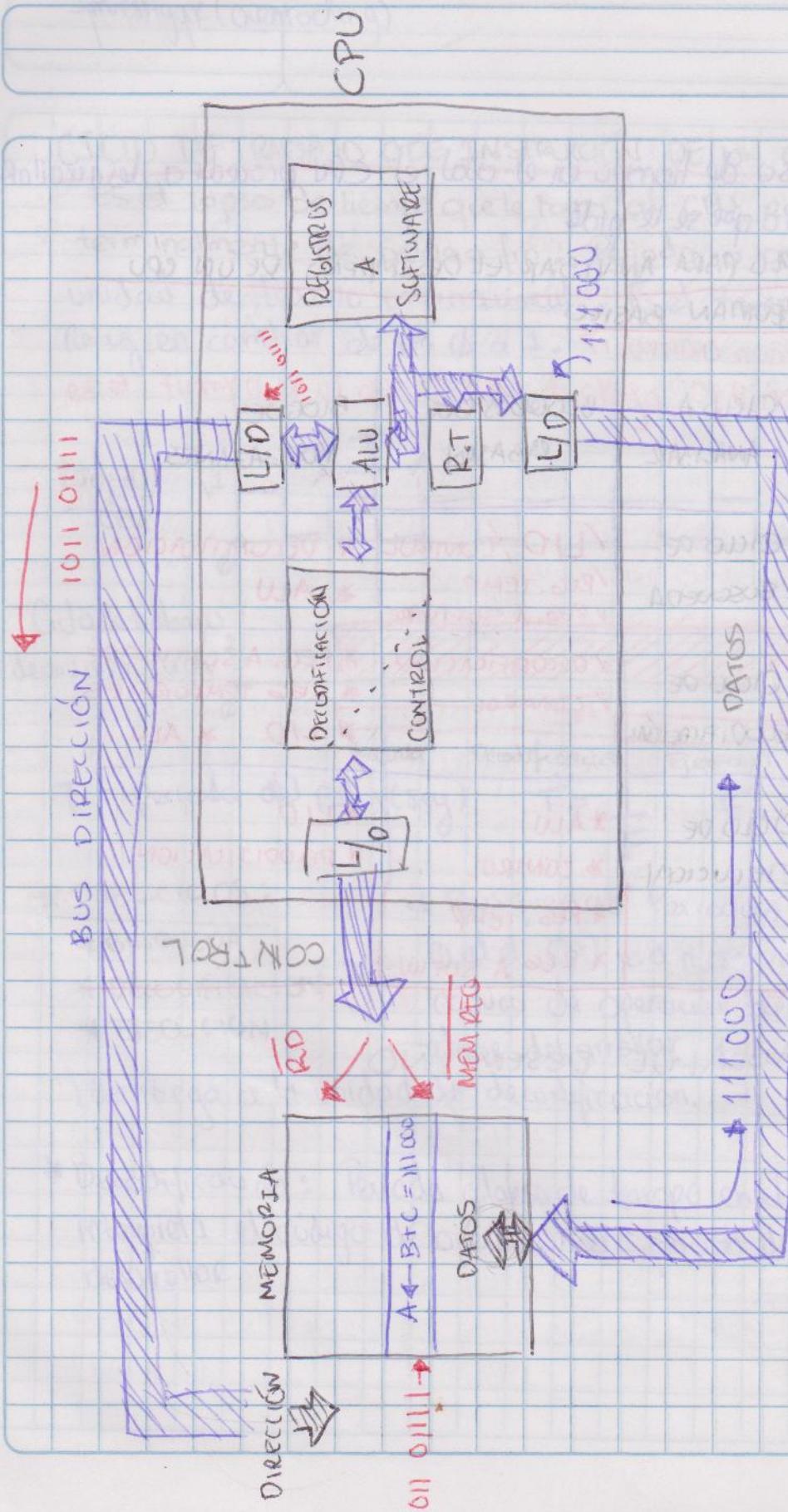
\* Plowación: lapso de tiempo en el cual el CPU procede a desarrollar terminativamente lo que se le pide

### TABLA DE EJEMPLO PARA ANALIZAR EL DESEMPEÑO DE UN CPU

#### CISC / VON NEUMAN BÁSICO

ORDEN A ANALIZAR	CICLO A ANALIZAR	BLOQUES QUE TRABAJAN	BLOQUES HOLGABANES
A ← B+C DONDE: A,B,C SON REGISTROS DE ACCESO POR SOFTWARE	CICLO DE BUSQUEDA	/LIO, CONTROL ✓REG TEMP ✓REG A SOFTWARE	* DECODIFICACION * ALU
	CICLO DE DECODIFICACION	✓DECODIFICACION ✓CONTROL	* REG A SOFTWARE * REG TEMPORALES * LIO * ALU
	CICLO DE EJECUCION	* ALU * CONTROL * REG TEMP * REG A SOFTWARE	* LIO * DECODIFICACION

### TABLA DE DESEMPEÑO



\* Fotocopia de este CPU

\* Esquema

\* Dibujado a mano de búsqueda

## EXAMEN CORTO 1

- ① Generar una tabla de desempeño del CPU/CISC / VON NEUMAN básico ; mostrando para los 3 ciclos , los bloques digitales que trabajan y los que no . para la instrucción : A[4] - M[100]
- Donde : "A" es un registro de acceso por software  
"M" es una memoria RAM estática
- M[100] : Significa el acceso a la celda Número "100"  
existe en "M"
- ② Diferencia entre un CPU RISC y un CPU CISC // von neuman hardware
- ③ Cuanta memoria tiene capacidad de manejar un CPU con un bus de dato de 16 Bits , un bus de dirección de 20 Bits y un bus de control de 15 Bits .  $2^k = 2^{20}$

①	Busq	L/D Reg x set , temp Control	Cod, ALU
	cod.	Cod , control Reg. temp	AW , L/D Reg. acc x soft
	ejec.	L/D, control Reg. acc x soft	ALU, cod
②	(1) # de Registros ; (2) Instrucciones ; set de instrucciones abajo nivel etc..		
	(3) Tienen un lote Acc. x soft limitado (RISC)		
③	Solo el bus de Dirección tiene la capacidad de decir cuantas celdas se van a manejar // $2^{20} = 1 \text{ MB}$ o un millón de bytes cada celda tiene 8 bits		

Memoria caché: acelera la velocidad del ordenador

Ciclo de Trabajo de un CPU (CT)

$$CT = \text{CICLO BUSQ} + \text{CICLO DE DECODI} + \text{CICLO EJECUCIÓN}$$

= 1T

Línea de Tiempo de desempeño CPU 80286

ORDENES A ANALIZAR: -oo (pasado)

- ①  $A \leftarrow A+B$  UNIDAD DE BUS
- ②  $C \leftarrow C \oplus A$  UNIDAD DE INSTRUCCION
- ③  $M[1000] \leftarrow C$  UNIDAD DE EJECUCION

→ oo (futuro)

UNIDAD DE BUS	UNIDAD DE INSTRUCCION	UNIDAD DE EJECUCION
Busca	Decodif.	Decodif.
NE	Decodif.	Decodif.
BUS	Decodif.	Ejecución

Donde:

A, B, C, son registros

instalados en la  
unidad de ejecución

M Memoria RAM  
estática

$M[1000]$  significa una  
incisión a la  
celda 1000m  
indexada.

1 T	1 T	1 T	1 T	1 T	1 T	1 T	1 T	1 T	1 T
Busca	Busca	Busca	Busca	Busca	Busca	Envío a memoria			
NE	Decodif.	Decodif.	Decodif.	Decodif.	Decodif.				
BUS						Finalización de cada			

Intel realiza los 3 en un  
periodo de tiempo

## Organización de los registros en los CPU Intel del 286 al Pentium 2

Hay 4 lotes de registros en estos micros (1) Lotes de registro de Propósito General (II II II II Indices y apuntadores) (2) (II II II de Segmento) (II IIII II de Mantenimiento)

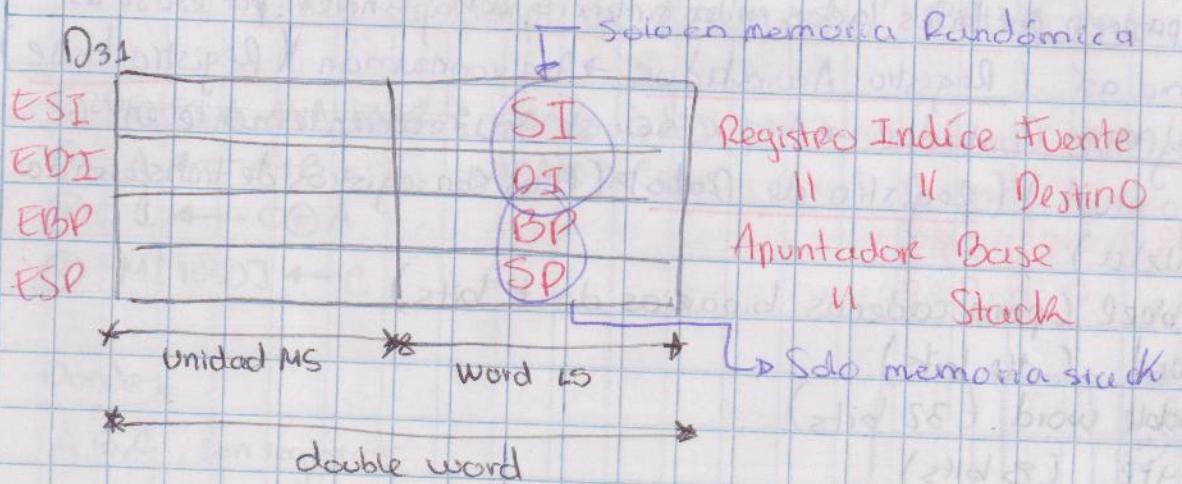
(1) Aparecen por todos lados en la sintaxis a bajo nivel, por eso se les llama así (Registro Acumulador → Arg von Neuman X Registro base) (Registro Contador → Arg CISIC, se usa recurrentemente en bajo nivel) (Registro de Dato) (Ellos son registros de transferencia paralela)  
nibble (para cadenas binarias de 4 bits)  
word (16 bits)  
double word (32 bits)  
byte (8 bits)

## Registros de Propósito General

	D31	D16	D8	D7	Do	
EAX		High AH	AH	AX	AL	Registro Acumulador
EBX			BH	BX	BL	Registro Base
ECX			CH	CX	CL	Registro Contador
EDX			DH	DX	DL	Registro Dato, index de puerto

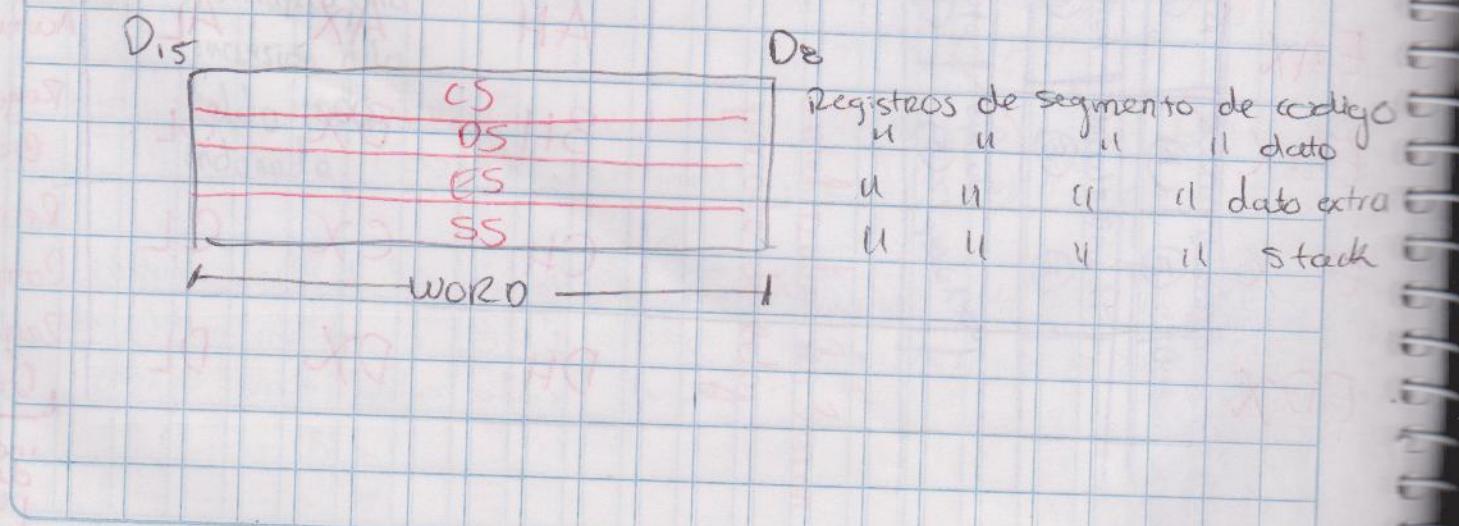
## Registros de Índices y apuntadores

- (A) → índices trabajan con memorias aleatorias      índice destino (1)  
(B) → apuntadores      índice fuente (2)  
                ||      ||      ||  
                estáticas      base (Random) (3)  
                ||  
                Stack (LIFO) (4)



## Registros de Segmento

- modo protegido (Registros de Sectores)  
→ " real (" " " " Segmentos)



{ PC = Contador de programa } → Le dice al CPU la celda donde tiene q buscar el próximo dato

(IP)

→ "Timón del CPU"

Contador Síncronico del Relej

búsqueda de la memoria

### Registros de Mantenimiento

	D31	D16-D8	D0	
EIP			IP	Registro Apuntador de Instrucciones
EF			F	Registro Status o bandera

Registro

(F) → Se maneja en forma paralela ; es el encargado de guardar el status de la última operación Aritmética o Lógica que el CPU acaba de realizar, más algún otro bit que lleva al contexto. Indica como está la máquina.

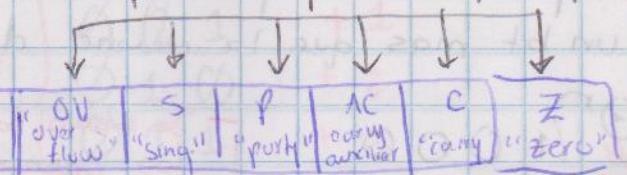
(Estudiar Complemento A2)

### REGISTROS BANDEIRAS

BANDEIRAS ASOCIADAS :

---	D	I	T
---	"Director"	"Interrupt"	"Trap"

Cosillas informáticas de tipo booleano soportados por un Flip-Flop



los más importantes para algoritmos a bajo nivel  
Son registros del status del resultado de la última operación realizada por el CPU.

\* Z "Zero" → Zero  
↳ Non Zero

\* C "Carry" → Carry  
↳ Non carry

\* P "Parity" → Parity Par  
↳ Parity Impar

50 +

20

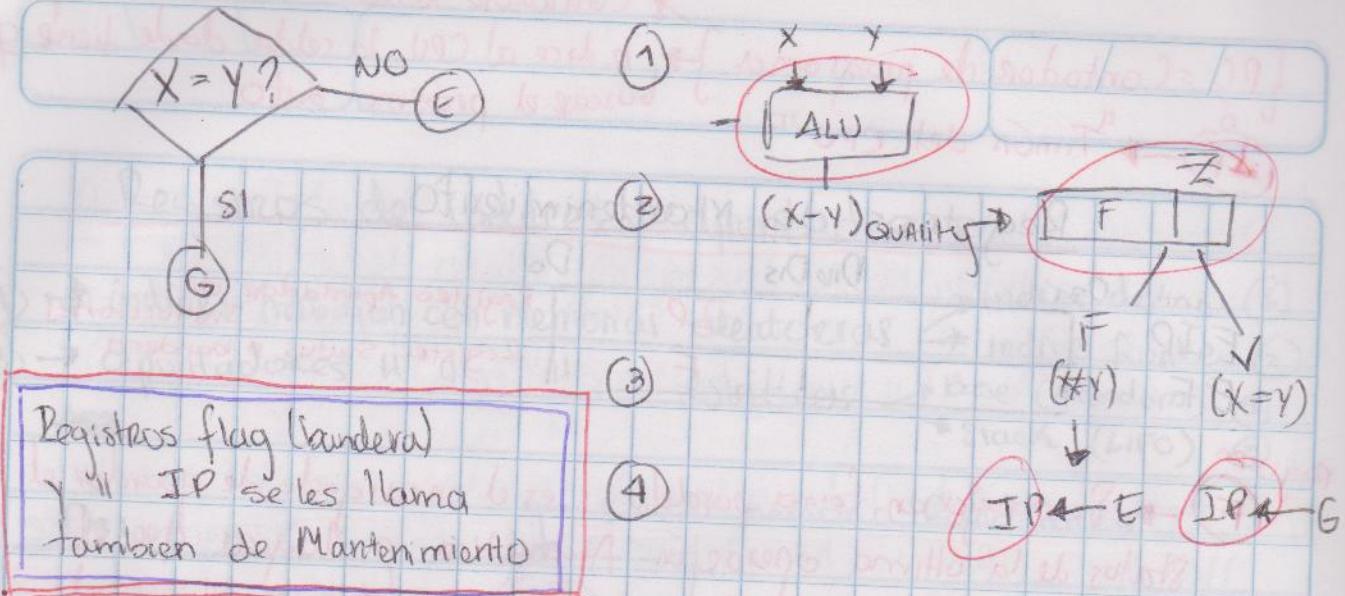
70

El registro bandera guarda los atributos del resultado

C > P

Para usar el registro bandera tiene que ser atraves de una operación aritmética o lógica.

# ALU, IP, Reg. Bandera



**Registros flag (bandera)**  
 y // IP se les llama  
 tambien de Mantenimiento

**Bandera Zero** → Se activa Si el resultado es un  $\phi$  total,  
 evaluado sobre la cadena binaria de los operandos involucrados

**Bandera Carry** → Se activa cuando al CPU se le hacee imposible  
 seguir manejando la cadena resultado por que supero el  
 tamaño de la cadena de los operandos, ~~es~~ cuando  
 despues de la operacion ARITMETICA el resultado posee  
 un bit mas que la cadena de los operandos.

Carry TRUE

$$\begin{array}{r} 1000 \\ + 1000 \\ \hline 0000 \end{array}$$

Se evalua la misma cantidad que de los operandos

Bandera Zero → Zero = TRUE  
 Non Zero = FALSE

Intel trabaja sus sumas y restas con el algoritmo de la base 10  
 no usa complemento A2

**Bandera Carry Auxiliar:** Se activa despues de una op. aritm. de suma o resta se genera un carry del bit 3 al 4

**Bandera Parity:** Se activa cuando el # de 1 que hay en la cadena es un numero par

Banderas OY y S: Cuando hay números con signo, bajo la condición de la convención del complemento A2

Complemento A2: Sacar complemento A1 y sumar 1 al bit menos significativo

0 en el bit más significativo = POSITIVO

1 = NEGATIVO

Banderas Signo y Overflow

Complemento A DOS: (convención con Signo)

Números sin Signo:

0 0 0  
0 0 1  
0 1 0  
0 1 1  
1 0 0  
1 0 1  
1 1 0  
1 1 1

0  
1  
2  
3  
4  
5  
6  
7

+

0 0 0  
0 0 1  
0 1 0  
0 1 1  
1 0 0  
1 0 1  
1 1 0  
1 1 1

0  
+1  
+2  
+3  
  
-3  
-2  
-1

+1 = 0 0 1

A1 C(1) = 1 1 0

A2 C(2) = +1

-1 = 1 1 1

+2 = 0 1 0

C(1) = 1 0 1

C(2) = 1 +

-2 = 1 1 0

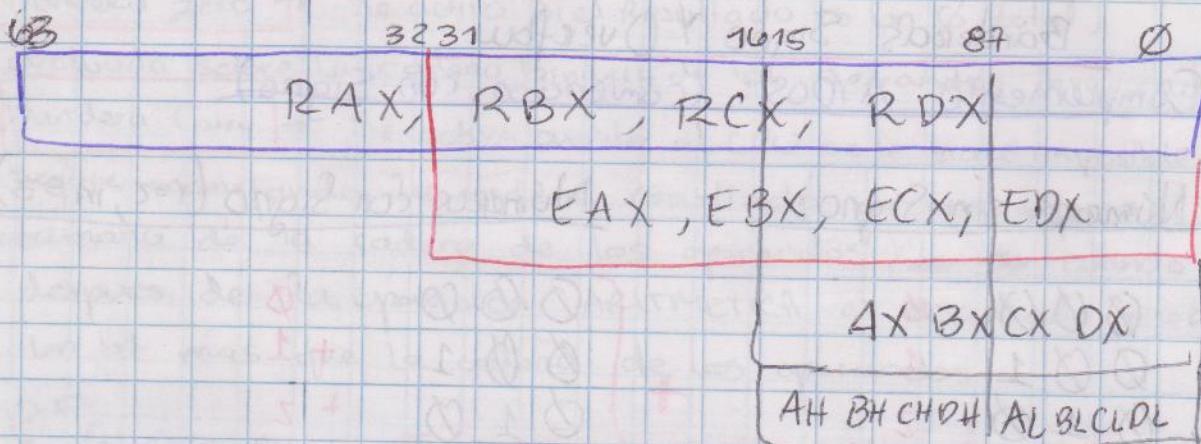
Ø 23  
111 / /

### Ejercicio:

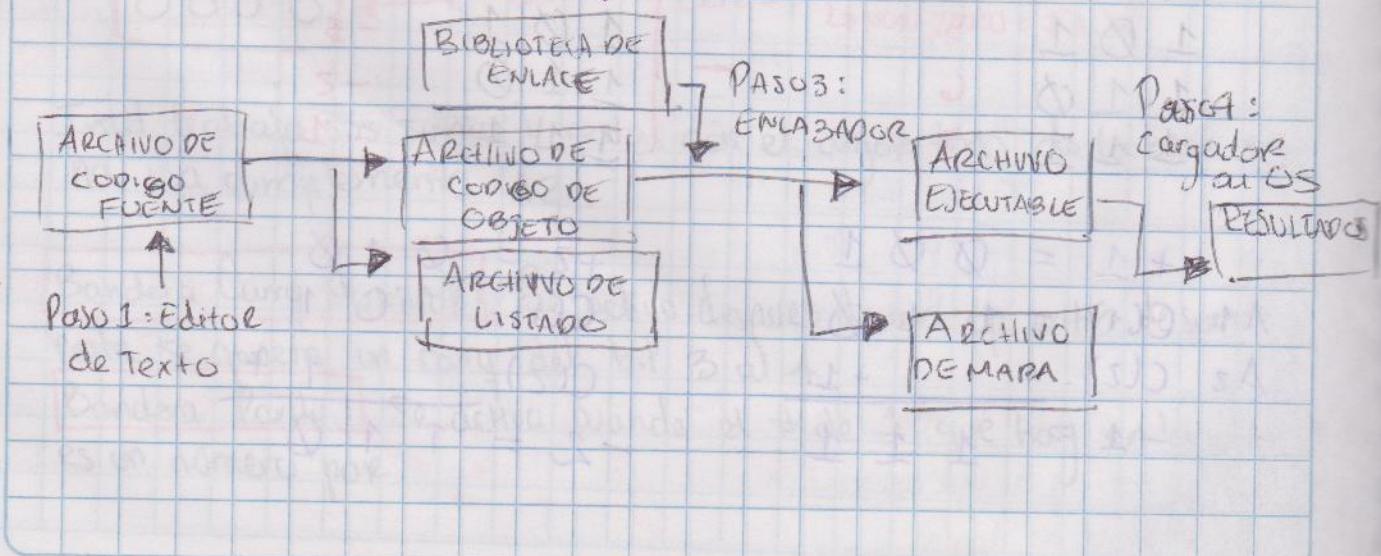
Indicar el valor lógico que reciben las banderas: OV, S, P, AC, C, Z después de efectuarse la operación.

$$A = B8$$

$$B = 5F +$$



Enlazador, encadenador, o Linker

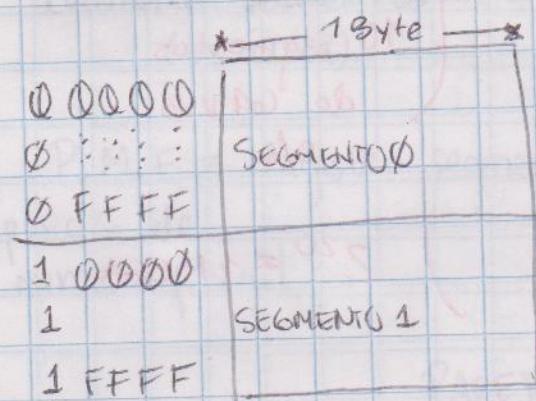


## MODO DE ADMINISTRACION DE MEMORIA POR SEGMENTACION (MODO REAL)

SEGMENTOS (CÓDIGO, DATO, DATO EXTRA, STACK)

Parafos = Conjunto de 16 celdas

1MB es la capacidad máxima que proporciona intel



SEGMENTO DE DATO → Se puede accesar de forma randomica, informacion miscelanea

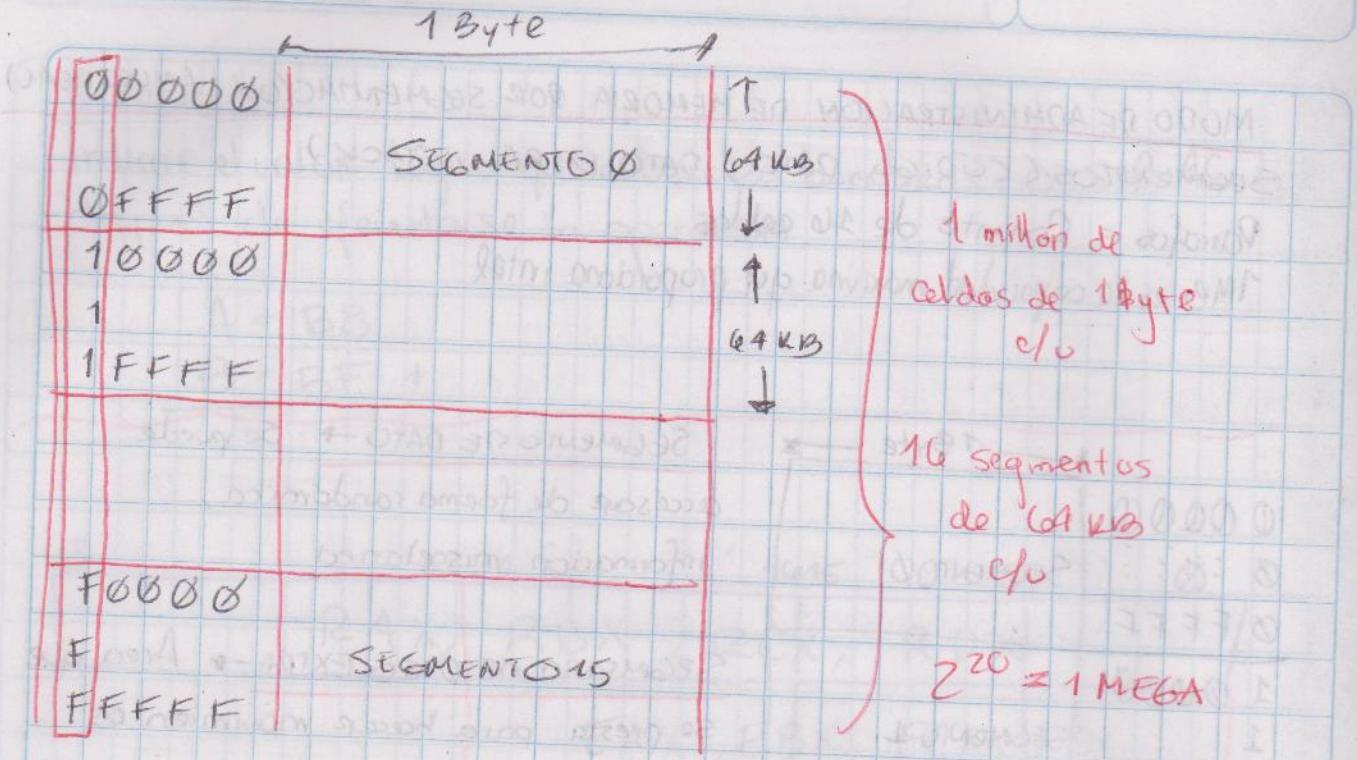
SEGMENTO DE DATO EXTRA → Area que se presta para hacer movimientos de datos

SEGMENTO DE CÓDIGO → Para asentor allí el ejecutable, comando y ordenes

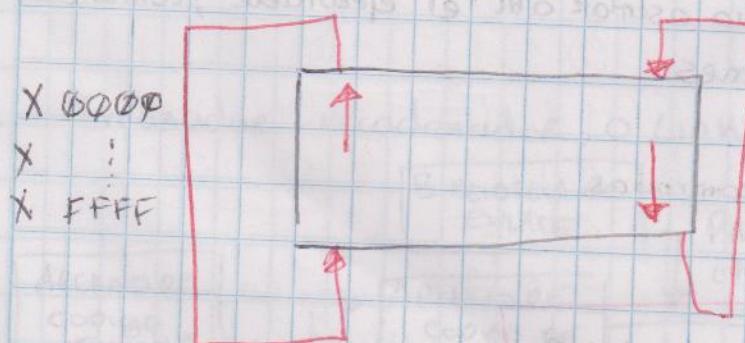
1 paraf = 16 celdas continuas

máximo = 64 KB

Área máxima de memoria = 1MB  
para trabajar



Con protección de fronteras en segmentos de 64 KB por cedididad



La cedididad se da solo para segmentos de 64 KB

20 nödeos que son los que se usan para la memoria

--	--	--	--	--

Existen 2 tipos de registros involucrados en la administración de memoria y específicamente en la identificación y navegación dentro de cada segmento.

### REGISTROS DE SEGMENTO

IDENTIFICA SEGUIMENTOS  
(ESTÁTICA)

### REGISTROS DE DESPLAZAMIENTO

NAVEGACIÓN EN SEGUIMENTOS  
(DINÁMICA)

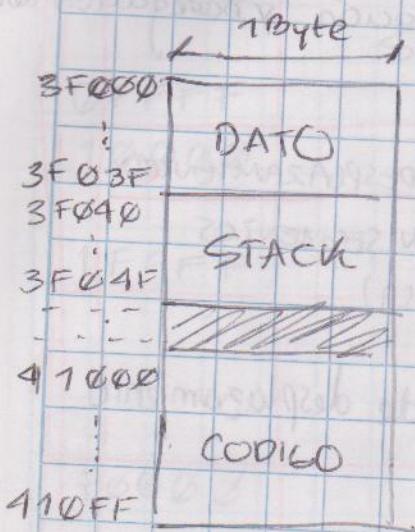
P.M.F = Registro segmento \* 10 + Registro de desplazamiento  
(posición de memoria física)

### PAREJAS ASIGNADAS

<u>PROPSITO USO</u>	<u>REGISTRO DE SEGMENTO</u>	<u>REGISTRO DE DESPLAZAMIENTO</u>
INSTALACIÓN DE CÓDIGO	CS	ID
DATOS	DS	DI, SI, BP, BX, NÚMEROS
MISCELÁNEOS		
TRANSFERENCIA	ES	DI
DATOS (STRING)		
STACK / LIFO	SS	SP

Cuando una aplicación uno asigna tamaños pequeños

### Ejemplo (MÓDULO TRASLADADO DE SEGMENTOS)



Obtenga los valores numéricos y cotas de los registros de segmento y desplazamiento, asociados a cada sección de memoria (vea dibujo adjunto). Indique también el número de celdas ó posiciones de memoria, que cada segmento contienen.

Posición de memoria física

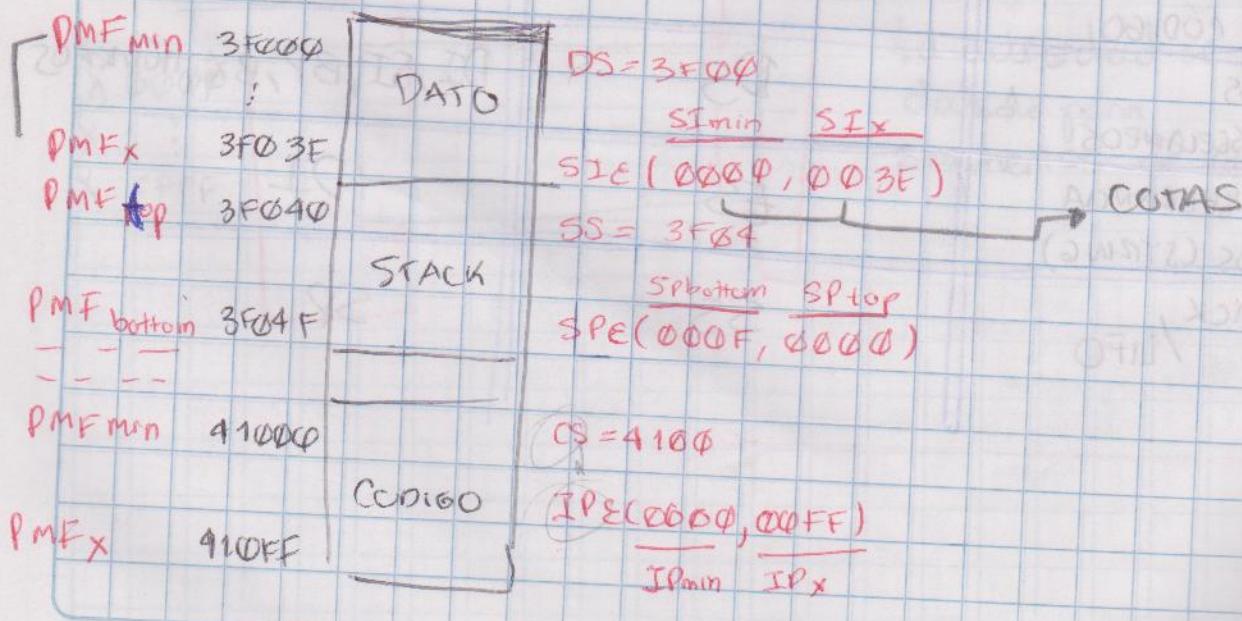
$$P.M.F = \text{Reg Segmento} * 10_H + \text{Reg Desplazamiento}$$

$$P.M.F_{\min} = DS * 10 + SI_{\min} \Rightarrow 3F0000 = 3F000 * 10 + SI_{\min}$$

$$\hookrightarrow SI_{\min} = 0000$$

$$P.M.F_x = DS * 10 + SI_x \Rightarrow 3F03F = 3F000 * 10 + SI_x$$

$$\hookrightarrow SI_x = 003F$$



### NÚMERO DE CELDAS TAMAÑO BYTE/SEGMENTO:

$$= \sum_{16-10} \{ DMF_x - DMF_{\min} \} + 1$$

DATO: Tamaño Bytes =  $\sum_{16-10} \{ 3F03F - 3F000 \} + 1$

$$\begin{aligned} T_B &= \sum_{16-10} \{ 003F \} + 1 = 0 \cdot 16^3 + 0 \cdot 16^2 + 3 \cdot 16^1 + 15 \cdot 16^0 + 1 \\ &= 64 \text{ celdas 1 Byte c/u (4 párrafos)} \end{aligned}$$

STACK:  $\sum_{16-10} \{ 3F04F - 3F040 \} + 1 = \sum_{16-16} \{ 0000F \} + 1$

$$= 0 \cdot 16^4 + 0 \cdot 16^3 + 0 \cdot 16^2 + 0 \cdot 16^1 + 15 \cdot 16^0 + 1 = 16 \text{ Bytes (1 párrafo)}$$

CÓDIGO:  $\sum_{16-10} \{ 410FF - 410000 \} + 1 = \sum_{16-16} \{ 000FF \} + 1$

$$= 0 \cdot 16^4 + 0 \cdot 16^3 + 0 \cdot 16^2 + 0 \cdot 15 \cdot 16^1 + 15 \cdot 16^0 + 1 = 256 \text{ bytes o 256 celdas de 1 byte c/u}$$

Cuando se da memoria uno ocupa bloques pequeños

Se busca el  $\oplus$  a la derecha luego se pita y se completan los 4 bits con  $\oplus$  ceros

Se elimina de derecha a izquierda 1er cero

5042 $\oplus$

5053F

5051 $\oplus$

51BFF

AB410 $\oplus$

ABAFF

DATO

Código

STACK

DS = 5042

SI min

SI x

SIEC  $\oplus$  00000

CS = 5051

SPEC  $\oplus$  0000

IP = 100F

SS = AB410

IP EC  $\oplus$  000FF

IP X = 00044

SI

IP

SP

Registros  
Segmento 0

CS = code Segment

DS = Data Segment

SS = Stack Segment

$$\text{Datos: } * 5042\oplus = 5042 \times 10 + SI_{min}$$

$$SI_{min} = 0000$$

$$* 5053F = 5042 \times 10 + SI_x$$

$$SI_x = 001F$$

$$\text{Código: } * P_{m_2} 5051\oplus = 5051 \times 10 + IP_{min}$$

$$IP_{min} = 00000$$

$$* 51BFF = 5051 \times 10 + IP_x$$

$$IP_x = 010BF$$

$$\text{Stack: } ABAFF = AB410 \times 10 + SP_{bottom}$$

$$SP_{bottom} = 000EF$$

9830

985011

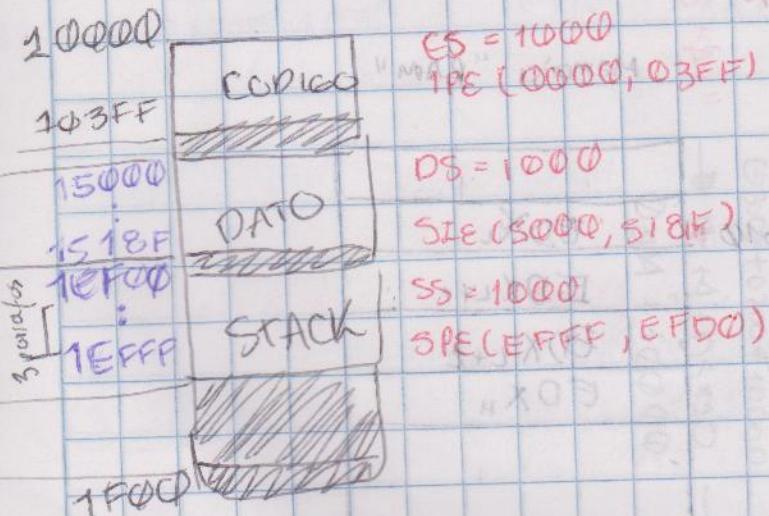
SP top = 0000

SUPONGA QUE POSEE UNA ÁREA LIBRE DE RAM, COMPRENDIENDA DE LA POSICIÓN :  $1000H \rightarrow 1F000H$  y proceda a instalar en ella 3 segmentos, uno de código de 1 Kbytes, uno de stack de 3 párrafos y uno de dato de 100 bytes. Deje constancia de eso a través de un dibujo, anotando los valores numéricos y cotas de los registros involucrados en el manejo de cada segmento.

$$1KB = 0000 \rightarrow C03FF$$

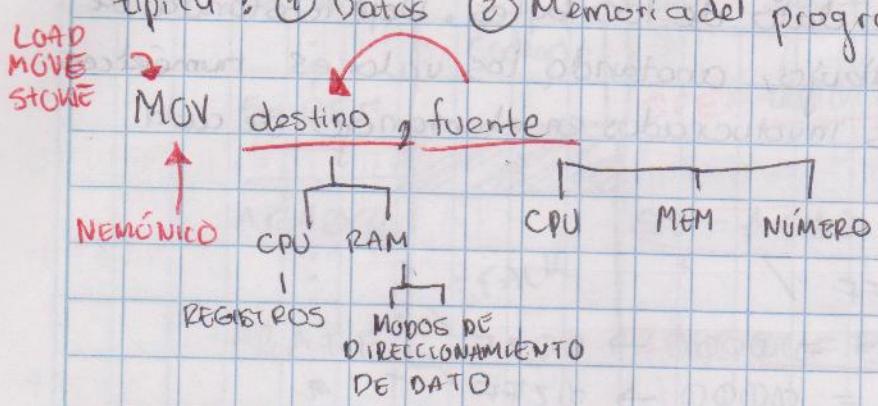
$$3 \text{ párrafos} = 18 \text{ bytes} = 0000 \rightarrow 002E$$

$$400 \text{ bytes} \approx 500 \text{ bytes} = 0000 \rightarrow 02FF$$



## Modos de Direccionamiento

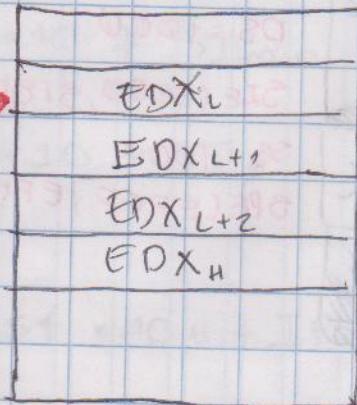
Son técnicas de clasificación de software a bajo nivel pero no es típica : ① Datos ② Memoria del programa ③ memoria pila



MOV EBX, BB0005610,  
MOV [EBX], EDX

BB0005610

Memoria "RAM"



## MODO DE DIRECCIONAMIENTO DE DATO

$$DS = 1000H | BX = 0300H | SI = 0200H$$

### SUB-MODOS:

- \* REGISTRO
- \* INMEDIATO
- \* DIRECTO
- \* INDIRECTO POR REGISTRO

MOV AX, BX  
MOV BL, 3AH  
MOV [1234H], AX  
MOV [BX], AX

REGISTRO "AX"  
REGISTRO BX  
REGISTRO "BL"  
REGISTRO "AX"

$1234 + 1000 \times 10$   
 $\rightarrow BX + DS \times 10$   
 $0300 + 10000$

$10300 \rightarrow$  MEMORIA  $10300H$   
 $10300H$

$10500H$   
MEMORIA

- \* BASE MAS INDICE

MOV [BX + SI], AX

REGISTRO "AX"

$BX + SI + DS \times 10$

$10300 + 10000$

$10300H$

- \* RELATIVO POR INDISTRO

MOV [BX + 4], AX

REGISTRO "AX"

$BX + 4 + DS \times 10$

$10300 + 10000$

$10300H$

Suma algebraica  
Número con signo

MOV [BX + 4], AX  
REGISTRO "AX"

$10300 + 10000$

$10300H$

$10300H$

## REGISTROS:

① DE PROPOSITO GENERAL

② DE PROPOSITO ESPECIFICO

① EN EL CPU DE 16 BITS

AX = ACUMULADOR (i, j, k, etc)

BX = BASE

CX = CONTADOR

DX = DATOS

Los 4 registros son de 16 bits

AX  
AH AL  
16 84 0

UNIDAD 8: MODO DE DIRECCIONAMIENTO DE DATO RELATIVO A LA BASE MAS INDICE

TIPO

INSTRUCCIÓN

FUENTE

GENERACION

DE LA DIRECCION

RELATIVO A LA

MOV Array[Bx+SI], AX

REGISTRO

Bx+SI + ARRAY + DS\*20

BASE MAS

INDICE

"AX"

0300 + 0200 + 50 + 100000

FORMA PRIMITIVA

MOV[Bx+SI+Offset], AX

Offset = Array = # con signo

INSTRUCCION EN CODIGO PRIMITIVO DATO NIVEL

ARRAY EQU 0x50

MOV ARRAY[Bx+SI], AX

MODO DE DIRECCIONAMIENTO A LA MEMORIA DEL PROGRAMA

JUMP / BRANCH

(for, if, do)

Salto condicionado

Saltos NO condicionados

Relativo  
Todos los numeros son  
considerados con signo  
Suma o resta al IP

Absolute  
aplica la etiqueta  
(numero) al  
contador del  
IP

No evalua ninguna  
bandera

- Modo de direccionamiento a la memoria del programa

① Directo

Desarrollo

$$\begin{array}{l} CS \leftarrow CS_N \\ IP_A \leftarrow IP_N \end{array}$$

JMP CS<sub>N</sub>, IP<sub>N</sub>

GO TO

GOSUB  
(LINK)

Modo segmento

} "trasciende

segmento"

SALTO - INTERSEGMENTADO

Por que tiene la capacidad  
de ir a otros segmentos

② Relativo

2.1 JMP e

offset

(valor numérico con signo)

$$E = 8 \text{ bits}$$

$$E = 16 \text{ bits}$$

-127 — +127

-32K +32K

salto corto

salto cercano

2.2 J Flag, e

offset

Simple:

Compuesta

con signo

—

sin signo

③ INDIRECTO

CASE / SWITCH

JMP Registro

↓

Ax

JMP [Registro]

Desarrollo

Desarrollo

$$IP \leftarrow Ax$$

$$IP \leftarrow Cx$$

$$IP \leftarrow [DS \times 10 + BX]$$

$$IP \leftarrow [DS \times 10 + DI + 2]$$

## F1 DE PROPOSITO GENERAL

## F2 DE PROPOSITO ESPECIFICO

### MODO DE DIRECCIONAMIENTO A LA MEMORIA DE LA PUA

PUSH registro

16 bits 32 bits 64 bits

CALL etiqueta

RETURNA

POP registro